Kubernetes Default Service CIDR Reconfiguration in Local Kind Cluster

Provision Local Kind Cluster Running Kubernetes v1.33

Create Local Kind Cluster

Multiple Service CIDRs in Kubernetes

Starting with Kubernetes **v1.31**, clusters can be configured with multiple service CIDRs. This feature supports not only expanding the service IP address space but also fully reconfiguring or replacing existing ranges. It provides greater flexibility for managing service networking, especially in scenarios involving IP exhaustion or architectural changes.

Why Use Kubernetes v1.33 on Kind for Experiment

Kubernetes **v1.33** makes support for multiple and reconfigurable Service CIDRs stable and turns it on by default. That means it's now possible to change or extend the service IP range without extra feature gates or downtime.

To spin up a local Kind cluster with Kubernetes v1.33 and a custom service CIDR, the following commands set up the environment:

```
# Recommend using the latest kind version
% kind version
kind v0.29.0 go1.24.2 darwin/amd64

# Create a single-node kind cluster using the kubernetes v1.33 node image
% kind create cluster --name kind-kube-v1.33 --image kindest/node:v1.33.1@sha256:050072256b9a903bd914c0b2866828150cb229cea0efe5892e2b644d5dd3b34f
Creating cluster "kind-kube-v1.33" ...
 ✓ Ensuring node image (kindest/node:v1.33.1) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾

 # Verify Kubernetes cluster and version
 % kubectl get no
```

```
NAME                            STATUS   ROLES           AGE   VERSION
kind-kube-v1.33-control-plane   Ready    control-plane   63s   v1.33.1
```

Inspect Existing Service IP Configuration

```
# Initial Service Cluster IP Range
% kubectl get servicecidrs
NAME         CIDRS          AGE
kubernetes   10.96.0.0/16   6m58s

# Existing Services
% kubectl get svc -A
NAMESPACE     NAME         TYPE        CLUSTER-IP   EXTERNAL-IP
PORT(S)                   AGE
default       kubernetes   ClusterIP   10.96.0.1    <none>
443/TCP                   8m13s
kube-system   kube-dns     ClusterIP   10.96.0.10   <none>
53/UDP,53/TCP,9153/TCP   8m12s

# Allocated IP Addresses
% kubectl get ipaddresses
NAME         PARENTREF
10.96.0.1    services/default/kubernetes
10.96.0.10   services/kube-system/kube-dns
```

Introduce New Service CIDR

Start by introducing new service CIDR as a temporary target during the transition, making it easier to move existing services off the old range without causing disruptions.

Reconfiguration Goal

Initial Service CIDR: `10.96.0.0/16`

New Service CIDR: `100.96.0.0/16`

Create a new Service CIDR with the updated IP range using the following manifest. Ensure that the name of the new Service CIDR does not conflict with the original.

```yaml
apiVersion: networking.k8s.io/v1
kind: ServiceCIDR
metadata:
  finalizers:
  - networking.k8s.io/service-cidr-finalizer
  name: kubernetes-new
spec:
  cidrs:
  - 100.96.0.0/16
```

Apply the manifest and then verify the new Service CIDR as long as the original.

```
# Save the manifest to local, e.g. servicecidr_kubernetes_new.yaml
% kubectl apply -f servicecidr_kubernetes_new.yaml
servicecidr.networking.k8s.io/kubernetes-new created

# Both the original and new Service CIDRs are listed
% kubectl get servicecidrs
NAME             CIDRS           AGE
kubernetes       10.96.0.0/16    18m
kubernetes-new   100.96.0.0/16   9s
```

Mark Default Service CIDR for Deletion

Mark the kubernetes default Service CIDR for deletion (it will remain pending due to existing IPs and finalizers). This prevents new allocations from the old range.

```
# Mark the default Service CIDR for deletion
% kubectl delete servicecidr kubernetes --wait=false
servicecidr.networking.k8s.io "kubernetes" deleted

# Verify the default Service CIDR remains with status of terminating
% kubectl get servicecidr kubernetes -o yaml
apiVersion: networking.k8s.io/v1
kind: ServiceCIDR
metadata:
  creationTimestamp: "2025-07-30T03:07:06Z"
  deletionGracePeriodSeconds: 0
  deletionTimestamp: "2025-07-30T03:31:03Z"
  finalizers:
  - networking.k8s.io/service-cidr-finalizer
  name: kubernetes
  resourceVersion: "2365"
  uid: 58d5fe79-7728-41c5-8d46-21f7d22ca220
spec:
  cidrs:
  - 10.96.0.0/16
status:
  conditions:
  - lastTransitionTime: "2025-07-30T03:31:03Z"
    message: There are still IPAddresses referencing the ServiceCIDR, please remove
      them or create a new ServiceCIDR
    reason: Terminating
    status: "False"
    type: Ready
```

> ⚠ Do not remove the finalizer on the default ServiceCIDR to force deletion. It's managed by the API server, which will automatically recreate it with the original IP range.

Recreate Existing Services

Non-default services need to be deleted and recreated manually so they get IPs from the new Service CIDR. In this local kind cluster, only the kube-dns service in the kube-system namespace needs recreation.

The default Kubernetes service, however, is managed by the API server and will be automatically recreated by the control plane, so it should be left until the end of the reconfiguration process.

```
# All existing services including the default and non-default
% kubectl get svc -A
NAMESPACE     NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
default       kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP                  8m13s
kube-system   kube-dns     ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   8m12s

# Export kube-dns service manifest
% kubectl get svc kube-dns -n kube-system -o yaml > svc_kube-dns.yaml

# Delete kube-dns service from kube-system namespace
% kubectl delete svc kube-dns -n kube-system
service "kube-dns" deleted
```

```
# Update service cluster IP range in the manifest
apiVersion: v1
kind: Service
metadata:
```

```yaml
  annotations:
    prometheus.io/port: "9153"
    prometheus.io/scrape: "true"
  labels:
    k8s-app: kube-dns
    kubernetes.io/cluster-service: "true"
    kubernetes.io/name: CoreDNS
  name: kube-dns
  namespace: kube-system
spec:
  clusterIP: 10.96.0.10 <-- 100.96.0.10
  clusterIPs:
  - 10.96.0.10 <-- 100.96.0.10
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: dns
    port: 53
    protocol: UDP
    targetPort: 53
  - name: dns-tcp
    port: 53
    protocol: TCP
    targetPort: 53
  - name: metrics
    port: 9153
    protocol: TCP
    targetPort: 9153
  selector:
    k8s-app: kube-dns
  sessionAffinity: None
  type: ClusterIP
```

```
# Apply the updated kube-dns service manifest
% kubectl apply -f svc_kube-dns.yaml
service/kube-dns created

# Verify the cluster IP of the updated kube-dns service
% kubectl get svc -A
NAMESPACE     NAME         TYPE        CLUSTER-IP    EXTERNAL-IP
PORT(S)                  AGE
default       kubernetes   ClusterIP   10.96.0.1     <none>
443/TCP                  43m
kube-system   kube-dns     ClusterIP   100.96.0.10   <none>
53/UDP,53/TCP,9153/TCP   41s

# Verify the IP allocation for the updated kube-dns service
% kubectl get ipaddresses
NAME          PARENTREF
10.96.0.1     services/default/kubernetes
100.96.0.10   services/kube-system/kube-dns
```

The default Kubernetes service remains assigned its original IP within the original Service CIDR, while the updated kube-dns service receives the new static IP `100.96.0.10` within the new Service CIDR.

Update API Server and Controller

Restart the kube-apiserver and kube-controller-manager static pods on the control plane node to apply the new Service CIDR.

```
# Find the control plane node name
% kubectl get no
NAME                           STATUS   ROLES           AGE   VERSION
kind-kube-v1.33-control-plane  Ready    control-plane   49m   v1.33.1

```

```
# Access the control plane node container
% docker exec -it kind-kube-v1.33-control-plane bash

# All the following commands ran in kind control plane node as in root@kind-kube-v1

# Find old Service CIDR reference in kube-apiserver and kube-controller-manager
% cd /etc/kubernetes/manifests/
% grep "service-cluster-ip-range" *.yaml
kube-apiserver.yaml:    - --service-cluster-ip-range=10.96.0.0/16
kube-controller-manager.yaml:    - --service-cluster-ip-range=10.96.0.0/16

# Update kube-apiserver and kube-controller-manager with new Service CIDR
% sed -i "s/10.96.0.0/100.96.0.0/g" kube-*.yaml

# Verify the update
% grep "service-cluster-ip-range" *.yaml
kube-apiserver.yaml:    - --service-cluster-ip-range=100.96.0.0/16
kube-controller-manager.yaml:    - --service-cluster-ip-range=100.96.0.0/16

% exit
```

The kube-apiserver and kube-controller-manager static pods will automatically restart to apply the change.

```
meng.xu@Mengs-MacBook-Pro doc % kubectl get po -A
NAMESPACE            NAME                                                READY   STATUS    RESTARTS      AGE
kube-system          coredns-674b8bbfcf-9mcb9                            1/1     Running   0             55m
kube-system          coredns-674b8bbfcf-zfkmz                            1/1     Running   0             55m
kube-system          etcd-kind-kube-v1.33-control-plane                  1/1     Running   0             55m
kube-system          kindnet-tbh4q                                       1/1     Running   0             55m
kube-system          kube-apiserver-kind-kube-v1.33-control-plane        1/1     Running   0             66s
kube-system          kube-controller-manager-kind-kube-v1.33-control-plane   0/1     Pending   0             4s
kube-system          kube-proxy-qj479                                    1/1     Running   0             55m
kube-system          kube-scheduler-kind-kube-v1.33-control-plane        1/1     Running   1 (97s ago)   55m
local-path-storage   local-path-provisioner-7dc846544d-m6gg6             1/1     Running   0             55m
```

Verify Service CIDR Update

```
# Tail kube-apiserver logs to verify ClusterIP allocator for the new Service CDIR is created
% k logs -f kube-apiserver-kind-kube-v1.33-control-plane -n kube-system | grep "Service CIDR"
I0730 04:01:26.642332       1 cidrallocator.go:301] created ClusterIP allocator for Service CIDR 10.96.0.0/16
I0730 04:01:26.642376       1 cidrallocator.go:301] created ClusterIP allocator for Service CIDR 100.96.0.0/16
I0730 04:01:26.642383       1 cidrallocator.go:277] updated ClusterIP allocator for Service CIDR 10.96.0.0/16
I0730 04:01:26.642385       1 cidrallocator.go:277] updated ClusterIP allocator for Service CIDR 100.96.0.0/16
I0730 04:01:48.927216       1 cidrallocator.go:277] updated ClusterIP allocator for Service CIDR 10.96.0.0/16
I0730 04:01:48.927247       1 cidrallocator.go:277] updated ClusterIP allocator for Service CIDR 100.96.0.0/16
```

```
I0730 04:01:48.927266       1 cidrallocator.go:277] updated ClusterIP allocator for Service CIDR 10.96.0.0/16
I0730 04:01:48.927269       1 cidrallocator.go:277] updated ClusterIP allocator for Service CIDR 100.96.0.0/16

# Review Original Service CDIR References
% kubectl get svc -A
NAMESPACE     NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)                  AGE
default       kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP                  61m
kube-system   kube-dns     ClusterIP   100.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   18m

% kubectl get ipaddresses
NAME          PARENTREF
10.96.0.1     services/default/kubernetes
100.96.0.10   services/kube-system/kube-dns
```

The original Service CIDR remains because the default Kubernetes service still uses its IP from that range.

Recreate Default Kubernetes Service

Delete the `kubernetes.default` service to force new kube-apiserver recreation within the new Service CIDR by API Server.

```
# Export the default kubernetes service manifest
% kubectl get svc kubernetes -o yaml > svc_kubernetes.yaml

# Delete the default kubernetes service
% kubectl delete svc kubernetes
service "kubernetes" deleted

# Verify the allocated IP is released
% kubectl get ipaddresses
NAME          PARENTREF
100.96.0.10   services/kube-system/kube-dns
```

Remove Old Service IP Cache

Recreating the default Kubernetes service may fail with an error like:

```
E0730 04:47:37.974657       1 controller.go:163] "Unhandled Error" err="unable to sync kubernetes service: Service \"kubernetes\" is invalid: spec.clusterIPs: Invalid value: []string{\"100.96.0.1\"}: failed to allocate IP 100.96.0.1: the provided range does not match the current range" logger="UnhandledError"
```

This happens because the old IP allocation is still cached in etcd.

```
# Access the control plane node container
% docker exec -it kind-kube-v1.33-control-plane bash

# All the following commands ran in kind control plane node as in root@kind-kube-v1

# Install etcd client
% apt update && apt install -y etcd-client

# Query Service IP Cache
```

```
% ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key get /registry/ranges/serviceips --prefix
/registry/ranges/serviceips

# outputs:
k8s

v1RangeAllocation#

"*28B
     10.96.0.0/16"

# Backup etcd before deleting the cache
% cd ~
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key snapshot save backup.db
# Outputs:
{"level":"info","ts":1753851323.8634012,"caller":"snapshot/v3_snapshot.go:119","msg":"created temporary db file","path":"backup.db.part"}
{"level":"info","ts":"2025-07-30T04:55:23.870Z","caller":"clientv3/maintenance.go:200","msg":"opened snapshot stream; downloading"}
{"level":"info","ts":1753851323.8704772,"caller":"snapshot/v3_snapshot.go:127","msg":"fetching snapshot","endpoint":"https://127.0.0.1:2379"}
{"level":"info","ts":"2025-07-30T04:55:23.892Z","caller":"clientv3/maintenance.go:208","msg":"completed snapshot read; closing"}
{"level":"info","ts":1753851323.8954964,"caller":"snapshot/v3_snapshot.go:142","msg":"fetched snapshot","endpoint":"https://127.0.0.1:2379","size":"2.2 MB","took":0.031472083}
{"level":"info","ts":1753851323.895573,"caller":"snapshot/v3_snapshot.go:152","msg":"saved","path":"backup.db"}
Snapshot saved at backup.db

# Delete Service IP Cache
% #  ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key del /registry/ranges/serviceips
#Outputs: 1

# Restart API Server
$ crictl ps | grep kube-apiserver

85560a56603f7       9a2b7cf4f8540       59 minutes ago      Running kube-apiserver            0                   49045760d6427 kube-apiserver-kind-kube-v1.33-control-plane            kube-system

% crictl rm -f 85560a56603f7
85560a56603f7
85560a56603f7

% exit
```

```
% kubectl logs kube-apiserver-kind-kube-v1.33-control-plane -n kube-system | grep "default/kubernetes"
I0730 05:01:27.517603       1 alloc.go:328] "allocated clusterIPs" service="default/kubernetes" clusterIPs={"IPv4":"100.96.0.1"}

% kubectl get svc -A
NAMESPACE     NAME         TYPE        CLUSTER-IP    EXTERNAL-IP PORT(S)                  AGE
```

```
default       kubernetes   ClusterIP   100.96.0.1    <none>
443/TCP                  88s
kube-system   kube-dns     ClusterIP   100.96.0.10   <none>
53/UDP,53/TCP,9153/TCP   73m
```

Regenerate API Server Certificate

The IP addresses listed in the API server's certificate Subject Alternative Name (SAN) are the trusted addresses where clients inside the cluster securely connect to the API server. These usually include the default Kubernetes service IP, the control plane node IPs, localhost, and any other relevant IPs.

When the cluster's Service CIDR changes, the default Kubernetes service is assigned a new cluster IP within this updated range. Because clients (like pods and components) connect to the API server using this default service IP, the new IP must be included in the API server's certificate SAN.

```
# Review the current API Server certificate SANs
# Verify recreation
% docker exec -it kind-kube-v1.33-control-plane bash

# All the following commands ran in kind control plane node as in
root@kind-kube-v1

% openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text |
grep -A5 "Subject Alternative Name"
            X509v3 Subject Alternative Name:
                DNS:kind-kube-v1.33-control-plane, DNS:kubernetes,
DNS:kubernetes.default, DNS:kubernetes.default.svc,
DNS:kubernetes.default.svc.cluster.local, DNS:localhost, IP
Address:10.96.0.1, IP Address:172.18.0.4, IP Address:127.0.0.1
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        5f:38:10:f7:26:1c:37:49:0e:98:57:13:ab:85:f2:71:09:7b:
        c1:5e:5e:1e:16:cb:df:41:ca:c7:93:5d:8d:29:5d:1a:96:60:
```

The current default Kubernetes service cluster IP, `10.96.0.1`, belongs to the original Service CIDR. The new cluster IP will be `100.96.0.1` to match the updated range.

If the new default service IP is not added to the SAN, the TLS handshake will fail whenever clients try to connect to the API server using that IP. This is because the certificate no longer matches the IP clients are connecting to, causing certificate verification errors.

Therefore, regenerating the API server certificate to include the new default Kubernetes service IP is a necessary step after changing the Service CIDR. This ensures that all internal cluster communication continues smoothly and securely without TLS errors.

```
# Access the control plane node container
% docker exec -it kind-kube-v1.33-control-plane bash

# All the following commands ran in kind control plane node as in
root@kind-kube-v1

# Export Current Kubeadm Cluster Configuration
% cd ~
% kubectl get configmap kubeadm-config -n kube-system -o
jsonpath='{.data.ClusterConfiguration}' > kubeadm-config.yaml
```

You may need to install `vim` on the control plane node for updating the configuration.

```
apt update && apt install -y vim
```

```yaml
# Add new cluster IP of default Kubernetes service to the configuration
apiServer:
  certSANs:
  - localhost
  - 127.0.0.1
  - 100.96.0.1 <-- Add the new Cluster IP
  extraArgs:
  - name: runtime-config
    value: ""
apiVersion: kubeadm.k8s.io/v1beta4
caCertificateValidityPeriod: 87600h0m0s
certificateValidityPeriod: 8760h0m0s
certificatesDir: /etc/kubernetes/pki
clusterName: kind-kube-v1.33
controlPlaneEndpoint: kind-kube-v1.33-control-plane:6443
controllerManager:
  extraArgs:
  - name: enable-hostpath-provisioner
    value: "true"
dns: {}
encryptionAlgorithm: RSA-2048
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: registry.k8s.io
kind: ClusterConfiguration
kubernetesVersion: v1.33.1
networking:
  dnsDomain: cluster.local
  podSubnet: 10.244.0.0/16
  serviceSubnet: 10.96.0.0/16 <-- Update to the new Service CIDR 100.96.0.0/16
proxy: {}
scheduler: {}
```

```
# Backup current certificate folder before regenerating apiserver certificate
% cp -r /etc/kubernetes/pki /etc/kubernetes/pki.bak

# Remove the current apiserver certificate and key to force regeneration
% rm /etc/kubernetes/pki/apiserver.crt /etc/kubernetes/pki/apiserver.key

# Regenerate apiserver certificate via kubeadm
% kubeadm init phase certs apiserver --config kubeadm-config.yaml
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kind-kube-v1.33-control-plane kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local localhost] and IPs [100.96.0.1 172.18.0.4 127.0.0.1]

# Verify CertSANs
% openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text | grep -A5 "Subject Alternative Name"
            X509v3 Subject Alternative Name:
                DNS:kind-kube-v1.33-control-plane, DNS:kubernetes, DNS:kubernetes.default, DNS:kubernetes.default.svc, DNS:kubernetes.default.svc.cluster.local, DNS:localhost, IP Address:100.96.0.1, IP Address:172.18.0.4, IP Address:127.0.0.1
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        93:fc:e9:2a:96:5d:05:d2:77:f4:a5:e7:bf:19:d0:b4:f8:86:
        8e:6c:45:e6:65:9d:3d:14:d1:97:ba:62:7e:23:ed:ee:cd:6b:
```

Update kubelet Configuration

The kubelet uses its config to set up DNS in Pods. If `/var/lib/kubelet/config.yaml` isn't updated with the new Service CIDR, newly created Pods may get an incorrect `/etc/resolv.conf`, pointing to the old Cluster DNS IP. This causes service name resolution to fail for services using the new CIDR.

```
# Access the control plane node container
% docker exec -it kind-kube-v1.33-control-plane bash

# All the following commands ran in kind control plane node as in root@kind-kube-v1

# Find original Cluster DNS IP reference in kubelet configuration
# i.e. the updated kube-dns service cluster IP 100.96.0.10
% cd /var/lib/kubelet/
% cat config.yaml | grep -A2 "clusterDNS"
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local

# Update ClusterDNS resolution to the new Cluster DNS IP
% sed -i "s/10.96.0.10/100.96.0.10/g" config.yaml

# Verify the change
% cat config.yaml | grep -A2 "clusterDNS"
clusterDNS:
- 100.96.0.10
clusterDomain: cluster.local

# Restart kubelet
% systemctl restart kubelet

% exit
```

Restart CoreDNS to Reflect DNS Changes

```
% kubectl get deployment -n kube-system
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
coredns   2/2     2            2           129m

% kubectl -n kube-system rollout restart deployment coredns
deployment.apps/coredns restarted

% kubectl get pods -l k8s-app=kube-dns -n kube-system
NAME                       READY   STATUS    RESTARTS   AGE
coredns-66799dd96c-cw7qx   1/1     Running   0          14s
coredns-66799dd96c-wck8q   1/1     Running   0          14s
```

Verify Service Resolution with New Cluster IP Range

Test that DNS correctly resolves the Kubernetes default service to the new Cluster IP and that the API server is reachable via that IP.

```
% kubectl run -i --rm --restart=Never nginx --image=nginx -- curl -k https://kubernetes.default.svc.cluster.local/version
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   379  100   379    0     0  84072      0 --:--:-- --:--:-- --:--:-- 94750
{
  "major": "1",
  "minor": "33",
  "emulationMajor": "1",
  "emulationMinor": "33",
```

```
  "minCompatibilityMajor": "1",
  "minCompatibilityMinor": "32",
  "gitVersion": "v1.33.1",
  "gitCommit": "8adc0f041b8e7ad1d30e29cc59c6ae7a15e19828",
  "gitTreeState": "clean",
  "buildDate": "2025-05-15T08:19:08Z",
  "goVersion": "go1.24.2",
  "compiler": "gc",
  "platform": "linux/arm64"
}pod "nginx" deleted
```

The test returned the API server version successfully, which confirms that service IP resolution and cluster networking are correctly updated with the new Service CIDR.

References

Kubernetes Default Service CIDR Reconfiguration